The Design and Implementation of a Hybrid Classical–Quantum Annealing Polar Decoder

Srikar Kasi[⋆,†], John Kaewell[†], Kyle Jamieson[⋆]

[⋆]Princeton University, [†]InterDigital, Inc.

***Abstract*—We present the Hybrid Polar Decoder (HyPD), a hybrid of classical CMOS and quantum annealing (QA) computational structures for decoding Polar error correction codes, which are becoming widespread in today's 5G and tomorrow's 6G networks. HyPD considers CMOS for the Polar code's binary tree traversal, and QA for executing a Quantum Polar Decoder (QPD)–a novel QA-based maximum likelihood submodule. Our QPD design efficiently transforms a Polar decoder into a quadratic polynomial optimization form amenable to the QA's optimization process. We experimentally evaluate HyPD on a state-of-the-art QA device with 5,627 qubits, for Polar codes of block length 1,024 bits, in Rayleigh fading channels. Our results show that HyPD outperforms successive cancellation list decoders of list size eight by half an order of bit error rate magnitude at 1 dB SNR. Further experimental studies address QA compute time at various code rates, and with increased QA qubit numbers.**

***Index Terms*—Polar codes, quantum computation, quantum annealing, channel decoding, cellular wireless networks**

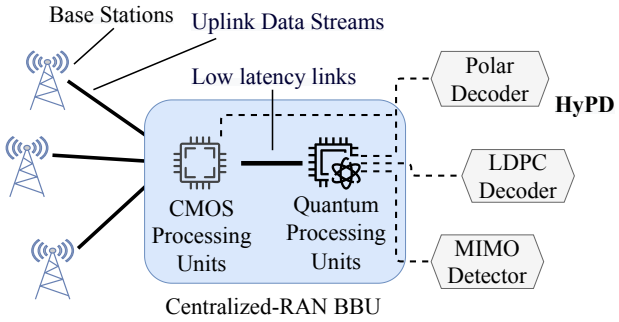


Figure 1: An envisioned **Hybrid Polar Decoder** (**HyPD**) deployment scenario in a centralized RAN (C-RAN) context, where a QA server augments a C-RAN baseband unit (BBU).

I. Introduction

The Polar Code, a type of error control code, was discovered in 2009 [1], and subsequently shown to have a number of desirable features: achievement of Shannon capacity for a wide range of channels, attainment of a low error floor (minimal bit error rate as a function of background noise), and a simpler code construction process than other leading competitors, such as Low Density Parity Check (LDPC) codes. While promising, however, Polar codes face several practical challenges if they are to manage decoder design complexity while at the same time maintaining their capacity-achieving properties. Low-complexity *Successive Cancellation* (SC) algorithms can only achieve capacity on Polar codes that have been constructed *a priori* with knowledge of the communication channel, which is unfortunately impractical in a wireless network context where user mobility causes wireless channels to be largely unpredictable. Furthermore, SC algorithms are fundamentally serial in nature, leading to low throughput in the decoding process. The *Successive Cancellation List* (SCL) decoder can approach Shannon capacity, but at the price of high complexity and high latency [2], thus compromising Polar codes' advantage over LDPC codes in this regard. Like SC, SCL makes decisions sequentially, which means that its decoder latency also grows at least linearly with code block length.

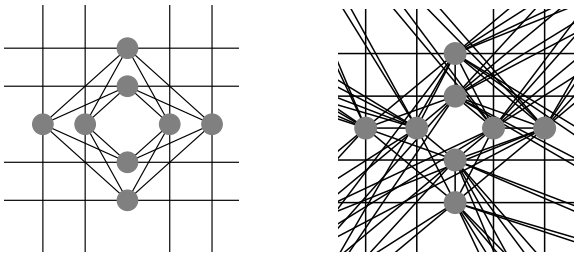
Consequently, Polar Code use in 5G New Radio [3] is limited to control channels with short block lengths, due to these considerations of processing throughput and latency surrounding the design of decoders discovered to date. But Polar codes' solid theoretical foundation, simple encoder implementation, and adjustable rate from zero to one would make them viable candidates for high speed data channels such as for 5G *enhanced Mobile Broad-Band* (eMBB), and for 5G *ultra-reliable, low-latency channels* (URLLC), if the latency of the decoder could be minimized while simultaneously maintaining a low bit error rate and near-capacity-limit rate performance. Overcoming processing throughput and latency limitations would also enable Polar codes' adoption in transformative 5G *Massive Machine Type Communications* (MMTC) technologies such as NB-IoT and LTE-M, which aim to scale cellular coverage densities to millions of devices per square kilometer, and hence put them on the 6G roadmap with more certainty.

It is with this vision in mind that this paper investigates a radically different processing architecture for a Polar Code decoder, one based on quantum annealing, to see if this emerging technology can potentially speed up the decoding of Polar codes using the fundamental, quantum-physical properties of superposition, entanglement, and tunneling. This would open up new possibilities in the design of the Polar Code decoder, thus overcoming the aforementioned hurdles to adoption. In our envisioned scenario, shown in Fig. 1, quantum processing units (QPUs) are co-located with CMOS processing units in a Centralized-RAN (C-RAN) data center, where quantum processors are used for heavyweight computational tasks in the cellular baseband unit (BBU), and CMOS processing units undertake lightweight computational tasks such as handling the network's control plane, transfer systems, and pre-/post-processing the QPU-specific computation [4], [5].

This paper presents the ***Hybrid Polar Decoder*** (***HyPD***), the first classical–quantum hybrid Polar decoder design that considers error correction decoding from the perspective of Quantum Annealing (QA). HyPD works by partitioning a long mother Polar code's binary tree into a number of shorter *sub-blocks*, where each sub-block is a largest perfect subtree with same leaf nodes as that of the mother Polar code (Fig. 4(a)). We structure HyPD's operation into classical and quantum processing modules. Our classical module considers CMOS hardware for the Polar code's binary tree traversal (*i.e.,* for updating log-likelihood ratios as in SC/SCL algorithms [2]), and it operates between the mother Polar code's root node and the root nodes of the partitioned sub-blocks. Our quantum module implements a Quantum Polar Decoder (QPD) on a QA device to solve these partitioned sub-blocks, decoding the transmitted bits. HyPD's classical and quantum modules exchange bit-likelihood and bit-decision information, respectively, back and forth in a feedback loop, until all bits are decoded.

Our quantum module's QPD is a novel maximum-likelihood Polar decoder design that efficiently formulates a one-to-one mapping between the Polar code's binary tree-structured encoder and a quadratic polynomial form that a QA can solve at the receiver, in order to decode the transmitted bits. This polynomial is a linear combination of multiple *cost penalty* functions we have created, which we refer to as *Node*, *Frozen*, and *Receiver* constraints. The Node and Frozen constraints work by raising a positive cost penalty to candidate bit strings that do not agree with the Polar encoding conditions, thus ensuring QPD outputs only valid bit strings. The Receiver constraints add a further cost penalty to all the bit strings whose magnitude depends on the proximity of an individual bit string to the received information (with channel noise), thus allowing QPD to select the most likely transmitted bit string. The QA returns the bit string with minimal cost penalty as its decoded solution. HyPD gathers the solution bit strings (corresponding to sub-blocks) returned by QPD and concatenates them, to output the final decoded codeword.

We experimentally evaluate HyPD on a state-of-the-art QA device with 5,627 qubits: Advantage_system4.1 [6], for CRC assisted Polar codes of block length 1,024 bits in multipath Rayleigh fading wireless channels at low signal-to-noise ratio (SNR) regimes of practical interest. Our evaluations consider BPSK modulation scheme and 200 data bits, which is typically the maximum payload size of uplink control information (UCI) in LTE and 5G-NR eMBB scenarios [7]. Results show that HyPD operating with eight-bit sub-blocks at 300 μs compute time outperforms SCL decoders operating at list size of eight by half an order of bit error rate (BER) magnitude at 1 dB SNR. Further studies present QA compute time analysis at various coding rates and with increased qubit numbers. While HyPD and QPD may in the future scale with noisy quantum devices' advances on their way to becoming fault-tolerant quantum computers, the lessons we have learned in the design and implementation of these decoders on QAs inform future quantum hardware design and RAN architecture evolution.



(a) Chimera Unit Cell **(b)** Pegasus Unit Cell

Figure 2: The figure shows *unit cell* structures of Chimera and Pegasus hardware topologies implemented in QA devices. Nodes are physical qubits, and edges are physical couplers.

II. Primer: Quantum Annealing

Quantum Annealing (QA) is an optimization based approach that aims to find the lowest energy *spin configuration* (*i.e.,* solution) of an *Ising model* described by the Hamiltonian:

$$H(s) = -A(s)H_I + B(s)H_P \tag{1}$$

$$H_I = \sum_i \sigma_i^x, \quad H_P = \sum_i h_i \sigma_i^z + \sum_{i<j} J_{ij}\sigma_i^z\sigma_j^z \tag{2}$$

where H_I is *initial Hamiltonian*, H_P is *problem Hamiltonian*, $s = t/t_a$ is called an *annealing schedule*, where t is time and t_a is *annealing time*, $\sigma_i^{x,z}$ are spin operators (Pauli matrices) acting on i^{th} qubit, h_i and J_{ij} are input problem parameters. A and B are two monotonic energy scaling signals such that at the start of an anneal (*i.e.,* $t = 0$), $A(0) >> B(0) \approx 0$ and at the end of an anneal (*i.e.,* $t = t_a$), $B(1) >> A(1) \approx 0$.

The annealing processor initializes qubits in a pre-known ground state of H_I, where each qubit is in an equal superposition state $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$, then gradually evolves this Hamiltonian from time $t = 0$ until $t = t_a$ by adiabatically introducing quantum fluctuations in a low temperature environment. This time-dependent evolution described by the Schroedinger Equation driven by these signals A and B is essentially the annealing algorithm. The system ideally stays in a low energy state and probabilistically reaches the global minima of the problem Hamiltonian H_P at its conclusion. The process of optimizing a problem in the QA is called *annealing*, and the time taken for annealing is called *annealing time* [6].

QA hardware is essentially comprised of two types of resources: *qubits* and *couplers*, organized regularly in groups of *unit cells*. Fig. 2 shows the unit cell structures of the *Chimera* and the *Pegasus* topologies implemented in recent and state-of-the-art QAs, respectively. QAs optimize Ising model problems, whose problem format matches the above problem Hamiltonian, described by the energy function $E = \sum_i h_i s_i + \sum_{i<j} J_{ij} s_i s_j$, where $s_i \in \{-1, +1\}$ is the i^{th} solution variable, h_i and J_{ij} are the input problem coefficients that the user supplies. The Ising problem format is equivalent to *quadratic unconstrained binary optimization (QUBO)* model, where the solution variables (q_i) take on values in $\{0, 1\}$, and it is obtained by the transformation $s_i \longrightarrow 2q_i - 1$, resulting

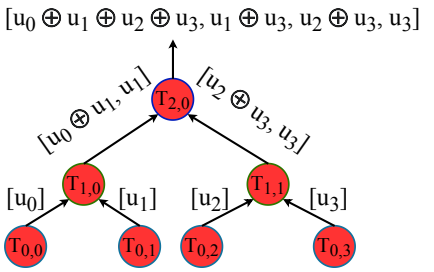


Figure 3: Encoding process of an example N = 4 Polar code. $T_{h,i}$ is a node of the tree at height h with index i. $\mathbf{x}_{ex}$ is the encoded codeword for the input vector $\mathbf{u}_{ex} = [u_0, u_1, u_2, u_3]$.

in the QUBO energy function $E_Q = \sum_i f_i q_i + \sum_{i<j} g_{ij} q_i q_j$, where f_i and g_{ij} are QUBO form coefficients of the problem. The QA probabilistically returns the solution variable configuration with minimum energy at its output.

III. Primer: Polar Codes

A binary $(N = 2^d, K)$ Polar code is described functionally by a generator matrix $\mathbf{G}_N = \mathbf{G}_2^{\otimes d}$, where $\otimes d$ operation represents d-fold Kronecker product, and $\mathbf{G}_2 = \left[\begin{smallmatrix} 1 & 0 \\ 1 & 1 \end{smallmatrix}\right]$ is called the *polarization kernel*. The *channel polarization* phenomenon in Polar codes is a transformation of N independent bits to N mutually interlinked bit-channels, where each bit-channel has its own probability of being decoded correctly (*i.e.,* reliability). The sequence of these bit-channels in sorted order of their reliabilities is called the *reliability sequence*. We now describe the Polar code encoding process [3].

Let N be the Polar code block length adapted for transmitting a message $\mathbf{m} = [m_0, m_1, ..., m_{K-1}]$ of length $K \leq N$ bits. The coding rate is then K/N. Using the reliability sequence,[1] construct the encoder *input vector* $\mathbf{u} = [u_0, u_1, ..., u_{N-1}]$ by assigning the message bits to K most reliable bit-channels, and set the remaining $N - K$ bits to a zero-value. The u_i's that are enforced to take a zero-value are called *frozen bits*. The encoded codeword is then $\mathbf{x} = \mathbf{u}\mathbf{G}_N$.

To visualize the QPD decoder (described later in §IV-B), we here demonstrate the encoder operation using a binary tree representation. Let us consider an example input vector $\mathbf{u}_{ex} = [u_0, u_1, u_2, u_3]$ of length $N = 4$ bits. Fig. 3 summarizes the $\mathbf{x}_{ex} = \mathbf{u}_{ex}\mathbf{G}_4$ computation of this encoding. Construct a perfect binary tree with N leaf nodes as shown in Fig. 3, and initialize each leaf node $T_{0,i}$ of the tree with bit u_i. Traversing the tree from height $h = 0$ (leaf) to $h = \log_2^N$ (root), each node $T_{h,i}$ $\forall h \in [1, \log_2^N]$ takes an input $[u_L|u_R]$ and generates an output $[u_L \oplus u_R|u_R]$, where u_L and u_R are the bit vectors of the left and right children of $T_{h,i}$ respectively, and $u_L \oplus u_R$ is a bit-wise XOR operation. The output vector obtained at the root is then the encoded codeword, which is then interleaved and transmitted over a wireless channel.

[1]This work adapts the reliability sequence of the 5G-NR standard [3].

IV. Design

In this section, we first detail HyPD's decoder operation (§IV-A) and then present the QPD's reduction of the Polar decoding problem into a QUBO form (§IV-B).

System Model. Let $\boldsymbol{u} = [u_0, u_1, ..., u_{N-1}]$ be the input vector corresponding to the Polar-encoded codeword $\boldsymbol{x} = [x_0, x_1, ..., x_{N-1}]$. Let $\boldsymbol{y} = [y_0, y_1, ..., y_{N-1}]$ be the respective received information with channel noise and interference. L_s be the log-likehood ratio (LLR) of bit s, $F(s,t)$ be the LLR of bit $s \oplus t$, and let $G(s,t,\widehat{s \oplus t})$ be the conditional LLR of bit t with respect to previously decoded bit $\widehat{s \oplus t}$. Then:

$$F(s,t) = 2\tanh^{-1}(\tanh L_s/2 \cdot \tanh L_t/2) \quad (3)$$

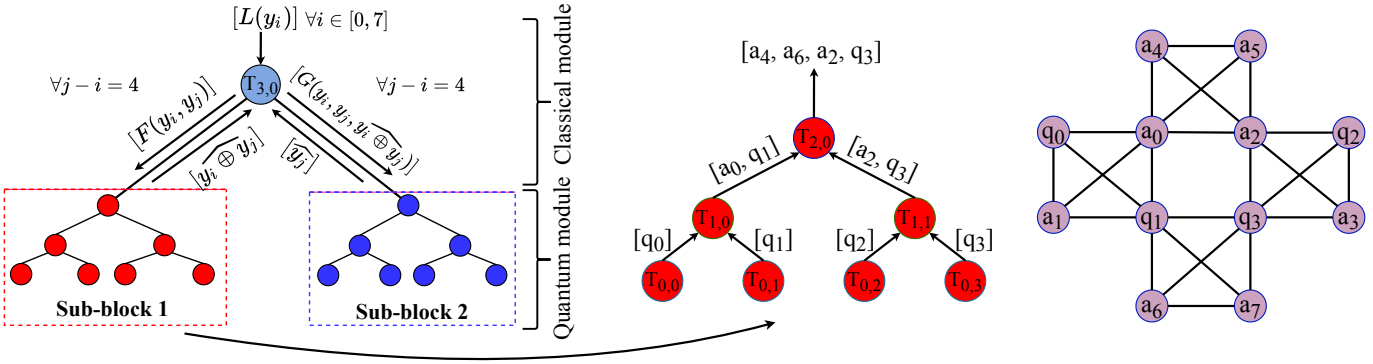
$$G(s,t,\widehat{s \oplus t}) = L_s(-1)^{\widehat{s \oplus t}} + L_t \quad (4)$$

A. HyPD: Hybrid Classical-Quantum Polar Decoder

Figure 4 summarizes the HyPD decoder operation. HyPD works by partitioning a long mother Polar code of block length N bits into N_{sub} shorter *sub-blocks*, where each sub-block is a largest perfect subtree with N_L leaf nodes of the mother Polar code tree. Figure 4(a) depicts this partitioning scheme for an example 8-bit Polar code with $N_{\text{sub}} = 2$ and $N_L = 4$.

We structure HyPD's decoding into classical and quantum processing modules. The classical module operates between the root node of the mother Polar code and the root nodes of sub-blocks as shown in Fig. 4(a), whereas the quantum module operates on these partitioned sub-blocks. HyPD's decoding begins at the root node by computing the LLRs of root node bits from the received soft data. Similar to SC/SCL decoder operation, we traverse the tree depth-first with priority given to the left branches [2]. Each node in our classical module sends to its left and right children the LLRs of their corresponding bits, by computing F and G functions, respectively. In this process, we obtain the bit LLRs of sub-blocks' root nodes. The bit LLRs corresponding to sub-blocks' root nodes are then our quantum module's input, using which we solve each sub-block on a QA device, and the solution obtained is fed back to classical module. This solution feedback is necessary for the classical module to compute F and G functions. Multiple solutions can be fed back to explore more decoding paths. This bit-likelihood and bit-decision information exchanges between our classical and quantum modules, respectively, represents our HyPD decoder operation. The decoder terminates when all the bits are decoded (*i.e.,* all sub-blocks are solved). We next demonstrate HyPD more fully with a running example.

Consider an 8-bit Polar code as in Fig. 4(a), and compute the LLRs of bits at the root node $T_{3,0}$ (*i.e.,* $[L(y_i)]$) using the received soft data. $T_{3,0}$ sends to root node of Sub-block 1 the LLRs of its corresponding bits by computing the F function (*i.e.,* $[F(y_i, y_j)]$). Sub-block 1 is then solved on QA and the solution obtained (*i.e.,* $[\widehat{y_i \oplus y_j}]$) is sent back to $T_{3,0}$. Using this solution, $T_{3,0}$ sends to root node of Sub-block 2 its corresponding conditional bit LLRs by computing the G function (*i.e.,* $[G(y_i, y_j, \widehat{y_i \oplus y_j})]$). Sub-block 2 is then solved on QA and the solution obtained (*i.e.,* $[\widehat{y_j}]$) is sent back to



(a) HyPD: 8-bit Polar code decoder **(b)** QPD: Sub-block 1 decoder **(c)** Sub-block 1 qubit connectivity

Figure 4: (a) HyPD decoder operation for an example 8-bit Polar code. **(b)** QPD's equivalent representation of Fig. 3 at the decoder (Bits in Fig. 3 are replaced by corresponding qubits). **(c)** QPD's qubit connectivity graph for Sub-block 1.

$T_{3,0}$. As all the sub-blocks are now solved, HyPD decoder terminates and outputs the decoded answer (§IV-B). While it is possible to perform the entire decoding using only the quantum module, the limited number of qubits in today's QA devices support for block lengths of up to only 128 bits. Therefore, we consider a hybrid classical–quantum approach to enable decoding of longer block lengths used in practice. We next demonstrate the sub-block decoding process on the QA.

B. QPD: Quantum Polar Decoder

Let $\mathbf{q} = [q_0, q_1, \ldots, q_{N_L-1}]$ be the *solution* qubits used to extract the input bits $\mathbf{u} = [u_0, u_1, \ldots, u_{N_L-1}]$ respectively. Let $\mathcal{F}$ be the set of all the frozen bits, and $\mathcal{T}$ be the set of all nodes in the Polar code's binary tree. Let any a_i be an *ancillary* qubit used for calculation purposes. Any b_i is a generic binary variable (*i.e.,* b_i can be any solution qubit or ancillary qubit).

1) QPD's QUBO Formulation: Our objective QUBO function comprises multiple terms, classified into three types: *Node*, *Frozen*, and *Receiver* constraints. The Node constraints (C_N) ensures that a candidate decoding agree with the Polar encoding conditions. If a candidate decoding violates these constraints, a cost penalty is raised for that candidate (*i.e.,* the candidate is raised in energy). The Frozen constraints (C_F) ensures all a candidate decoding agree with the frozen bit conditions (*i.e.,* qubits that represent frozen bits must take a zero value). If a candidate decoding disagrees, a cost penalty is raised for that candidate. The Receiver constraints (C_R) introduce a further cost penalty to all valid candidates, whose magnitude depends on the proximity of an individual candidate to the received information. They thus encourage the decoder to find the decoding that most closely matches the received information. The entire QUBO objective function is a weighted linear combination of these cost functions:

$$\arg\min_{\mathbf{q}} \left\{ W_N \sum_{\forall T \in \mathcal{T}} C_N(T) + W_F \sum_{\forall u_i \in \mathcal{F}} C_F(q_i) + W_R \sum_{\forall j} C_R(b_j) \right\} \quad (5)$$

The positive weights W_N, W_F, and W_R prioritize the Node, Frozen, and Receiver constraints respectively. Experimental evaluations show that $W_N = +1$, $W_F = +4$ per qubit, and $W_R = 2 - R_{sub}$ achieve good performance, where R_{sub} is the coding rate within the sub-block under consideration.

2) Node Constraints: From III, we observe that the Polar encoder performs only XOR operations. Let us define $\mathcal{E}_T$ as the set of all XOR operations the encoder performs at node T in the encoder binary tree representation. For each $T \in \mathcal{T}$ (defined earlier in §IV-B), we define a *Node* constraint as:

$$C_N(T) = \sum_{\forall \text{XOR}(b_i, b_j) \in \mathcal{E}_T} (b_i + b_j - a_k - 2a_{k+1})^2 \quad (6)$$

where b_i and b_j represent the qubits whose equivalent bits are XORed at node T in the encoding process, and a_k, a_{k+1} are ancillary qubits. The value of $k \in \{2p | p \in \mathbb{W}\}$ is chosen such that each ancillary qubit is only introduced once. We observe that $C_N(T)$ is in sum-of-squares form, thus at the minimum energy (*i.e.,* $C_N(T) = 0$), the sum $b_i + b_j$ must be equal to the sum $a_k + 2a_{k+1}$. Since all the variables are binary, this implies that $a_k = b_i \oplus b_j$ in the minimum energy configuration. We next demonstrate the working process of Node constraints more fully with a running example.

Let us continue with Sub-block 1, whose encoder tree can be visualized as in Fig. 3 with input vector $\mathbf{u}_{ex} = [u_0, u_1, u_2, u_3]$ and encoded codeword $\mathbf{x}_{ex} = [x_0, x_1, x_2, x_3]$. Let $\mathbf{q}_{ex} = [q_0, q_1, q_2, q_3]$ be the solution qubits used at the decoder to extract the bits $[u_0, u_1, u_2, u_3]$ respectively. Figure 4 (b) shows QPD's equivalent representation of Fig. 3 using respective qubits at the decoder. In this example, $\mathcal{T} = \{T_{2,0}, T_{1,0}, T_{1,1}, T_{0,0}, T_{0,1}, T_{0,2}, T_{0,3}\}$, and the code height is two. Similar to encoding, we traverse the tree from leaf to root for constructing QPD's Node constraints as follows.

At height = 0, we note that the nodes $\{T_{0,0}, T_{0,1}, T_{0,2}, T_{0,3}\}$ perform no computation, and so the Node constraints of these nodes are zero (*i.e.,* $C_N(T_{0,i}) = 0\ \forall i$). At height = 1, we have two nodes $\{T_{1,0}, T_{1,1}\}$ that perform one XOR operation each (see Fig. 3). In particular, $T_{1,0}$ computes $u_0 \oplus u_1$ and $T_{1,1}$ computes $u_2 \oplus u_3$. Thus using Eq. 6, we construct two Node constraints as: $C_N(T_{1,0}) = (q_0 + q_1 - a_0 - 2a_1)^2$ and $C_N(T_{1,1}) = (q_2 + q_3 - a_2 - 2a_3)^2$. Here a_0 equals $q_0 \oplus q_1$, and a_2 equals $q_2 \oplus q_3$ in the minimum energy solution. At height = 2,

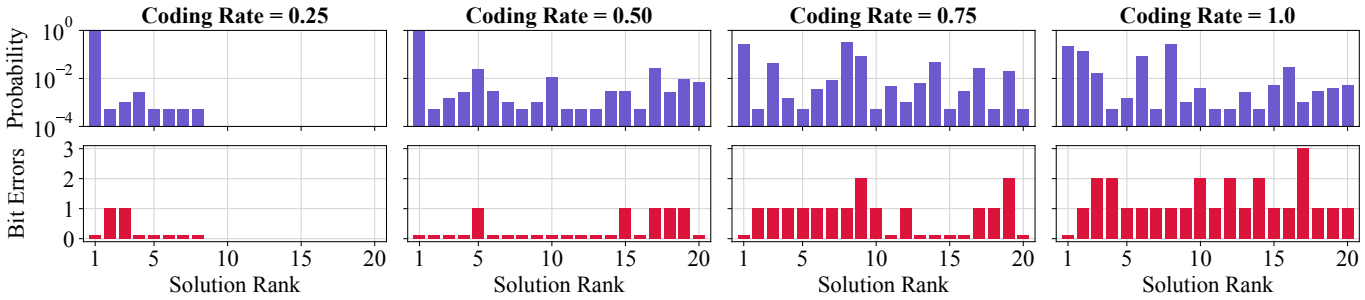


Figure 5: QPD's performance of partitioned sub-blocks at SNR 1dB, showing first 20 solutions. Sub-blocks with low coding rates achieve high Rank 1 solution probability. At coding rate 0.25, only eight distinct solutions are returned in $2,000$ anneals.

the root node $T_{2,0}$ performs two XOR operations (see Fig. 3): $u_0 \oplus u_1 \oplus u_2 \oplus u_3$, and $u_1 \oplus u_3$. We note that these computations are equivalent to $a_0 \oplus a_2$ and $q_1 \oplus q_3$ respectively. Hence, using Eq. 6 we construct the Node constraint: $C_N(T_{3,0}) = (a_0 + a_2 - a_4 - 2a_5)^2 + (q_1 + q_3 - a_6 - 2a_7)^2$. Here a_4 equals $a_0 \oplus a_2$, and a_6 equals $q_1 \oplus q_3$ in the minimum energy solution. The output vector obtained at the root node $[a_4, a_6, a_2, q_3]$ (see Fig. 4 (b)) is now bit-wise equivalent to $[x_0, x_1, x_2, x_3]$. We hence refer the output vector obtained at the root node to the *equivalent encoded codeword (EEC)*.

3) Frozen Constraints: From §III, we note that frozen bits do not carry user information and that they are always assigned zero value. Hence we define a *Frozen* constraint as:

$$C_F(q_i) = q_i \qquad \forall u_i \in \mathcal{F} \tag{7}$$

We observe that C_F is minimum when the qubits that represent frozen bits take a zero value (*i.e,* not one value).

4) Receiver Constraints: We next consider the *EEC* obtained from the Node constraints, and compute its closeness to the received information using a *Receiver* constraint as:

$$C_R(b_j) = (b_j - Pr(b_j = 1|\mathbf{y}))^2 \qquad \forall b_j \in EEC \tag{8}$$

where the probability $Pr(b_j = 1|\mathbf{y})$ can be computed for various modulations and channels, using the LLR information the HyPD's classical module supplies (§IV-A). We note that C_R is minimum for a $b_j \in \{0, 1\}$ that has a greater probability of being the corresponding bit at the encoder.

Fig. 4(c) depicts the qubit connectivity structure (*i.e.,* problem graph) of Sub-block 1, showing several four-cliques, where nodes and edges represent variables and quadratic terms of Eq. 5 respectively. Looking at Eq. 6, we see that the connectivity of these four-cliques mirrors the connectivity of QPD's Node constraints. In order to optimize a QUBO problem on a QA device, the first step is to specify the QUBO problem as an equivalent Ising problem (§II). Next step is to map this Ising problem onto the physical QA hardware, via a process known as *embedding*. We map QPD's problem graphs on to the physical QA hardware via a customized embedding scheme, where each Ising variable (*e.g.,* a node in Fig. 4(c)) is mapped on to eight physical qubits present in a unit cell, and each Ising coupler (*e.g.,* an edge in Fig. 4(c)) is mapped on to eight physical couplers connecting two unit cells. Heuristic tools such as *minorminer* [8] can also be used for embedding problems on QA, however, such heuristic embeddings generate irregular-length qubit chains which may degrade the solution quality [9]. At the end of QPD decoding, the EEC qubit configuration is fed back to HyPD's classical module. The final decoded answer of HyPD is the qubit configuration that represents user data (*i.e.,* $[q_i]$ $\forall u_i \notin \mathcal{F}$).

V. Implementation

We implement HyPD's classical module on 2.3 GHz eight-core Intel CPU with 14 nm CMOS process, and quantum module on 5,627-qubit Advantage_system4.1 QA. Our decoder targets 1,024-bit 5G-NR Polar codes with BPSK modulation and 200 message data bits, which is typically the maximum UCI payload in LTE and 5G-NR eMBB scenarios [7]. Our encoder implementation follows 5G-NR specifications [3]. In particular, a 11-bit CRC is attached to user data, frozen bits and sub-channels are allocated, and then the mother polar code encoding is performed as described in §III. This encoded data is next passed onto sub block and channel interleavers, and then transmitted over a wireless multipath Rayleigh fading channel. At the receiver, we de-interleave the received soft information accordingly and then perform HyPD's decoding.

Current QAs have a 4–40 μs coefficient programming time, 0–10 ms post-programming thermalization time, 25–150 μs solution readout time, and 0–10 ms post-readout thermalization time. Thermalization times are user-specified, and we set it equal to default 1 ms. These overhead times, however, can be reduced several orders of magnitude by system integration (see [5] for detailed analysis). In our particular QA device, there are 13 defective qubits, each in a different unit cell, and we use only 7 available physical qubits in such unit cells. Practical challenges include embedding, coefficient range and precision, and analog QA machine noise called *integrated control error* (ICE). ICE is caused by qubit flux-noise, quantization, among others [6], and it alters problem biases ($h_i \rightarrow h_i \pm \delta h_i$) and coupler strengths ($J_{ij} \rightarrow J_{ij} \pm \delta J_{ij}$). Although the errors δh_i and δJ_{ij} are currently on the order of 10^{-2}, these may disturb the solution quality in scenarios where ICE noise erases significant information from the problem. Nevertheless, we increase the solution quality via the standard method of

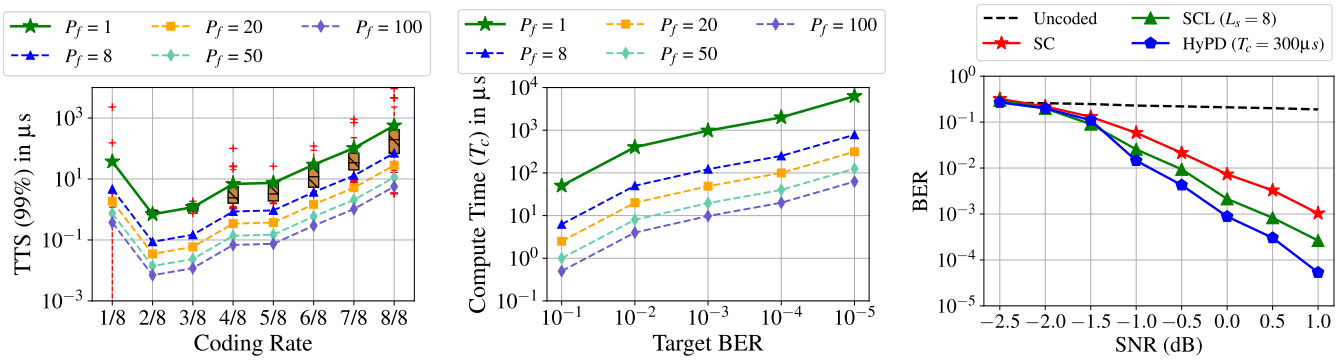


Figure 6: HyPD's end-to-end system performance. In the figure, boxes' lower/upper whiskers and quartiles represent $10^{th}/90^{th}$ and $25^{th}/75^{th}$ percentiles respectively. Line trends show averages, and P_f is QA parallelization factor.

running multiple anneals for a problem, where each anneal reads out the solution bit-string once. Current QAs support bias values in $[-4.0, +4.0]$, and coupler strength values in $[-2.0, +1.0]$. QPD's Ising form coupler strengths take values in $[-1.0, -0.5, +0.5, +1.0]$ only, falling into the supported range of QA, and we set embedding coupler strengths to -2.0. QPD's biases are quantized in steps of 1.5 value and shared among physical qubits in the embedding process, tuning them into the supported range of QA. Our end-to-end evaluation results capture all the sources of QA imprecision (*e.g.,* QA analog hardware noise and control error) [6].

VI. Evaluation

Let us define an *instance I* as a 1,024-bit Polar decoding problem. We partition each instance into 128 sub-blocks with eight bits each (as described in §IV-A). For each sub-block decoding, we perform $2,000$ anneals, where each anneal potentially returns a distinct solution to the problem due to the heuristic sampling nature of QA. If N_s^I is the number of distinct solutions returned for a problem, we rank these solutions in increasing order of their energies as $R_1, R_2, ..., R_{N_s^I}$, and note the solutions' bit errors and occurrence probabilities. Eight sub-blocks, each corresponding to a different instance, are parallelized in a single QA anneal, by mapping problems to distinct physical locations in the QA hardware. The minimum energy solutions of QPD's sub-blocks are fed back to HyPD's classical module (§IV-A).

A. QPD's Sub-block Performance

We first investigate in Fig. 5 the sub-block decoding performance of QPD. The figure shows that sub-blocks with low coding rates (0.25 and 0.5) achieve a high probability of finding the correct answer (*i.e.,* a Rank 1 solution), whereas sub-blocks with high coding rates (0.75 and 1.0) achieve a low correct answer probability. This is because at high coding rates, there are less frozen bits for error correction, making it difficult for the QA to solve the problems correctly. We further note that the number of bit errors is zero in the Rank 1 solutions at all coding rates. Solutions with higher rank (> 1) and zero bit errors imply that ancillary qubits, but not solution qubits that represent user data, are errored (§IV-B).

B. HyPD's System Performance

We next investigate in Fig. 6 the end-to-end system performance of HyPD. *Time-to-solution (TTS)* is a figure of merit for the QA performance, and TTS(P) represents the time required to reach the minimum energy solution with a probability P, computed as: TTS(P) = $T_a \cdot \log(1-P)/\log(1-P_1)$, where T_a is the annealing time and P_1 is the probability of R_1, the minimum energy solution. If P_f problems are parallelized, then the effective TTS of a problem is reduced by a factor P_f. In Fig. 6, curves corresponding to P_f = 8 represents current parallelization in a 5K-qubit QA device, whereas P_f of 20, 50, and 100 represent projected parallelization in near-term future QA devices with 14K, 35K, and 70K qubits respectively.

TTS Analysis. Fig. 6 (*Left*) shows sub-blocks' TTS(99%) performance with $T_a = 1\ \mu s$: TTS scales proportionally with sub-block's coding rate, reaching a worst-case 70 μs for sub-blocks with a 1.0 coding rate (P_f = 8). This is because at high coding rates, R_1 solution probability is low (§VI-A). With P_f = 100, this worst-case TTS reaches to 5.6 μs. We further note that TTS at coding rate of 1/8 deviates from the trend. This is because at very low coding rates, the energy gap between the minimum energy solution and the rest diminishes, making it difficult for QA to distinguish the minimum energy solution. To overcome this, techniques such as *quantum annealing correction* may be employed [10]—we leave for future work.

Compute Time. Fig. 6 (*Middle*) shows HyPD's compute time (T_c) requirements to achieve a target bit-error-rate (BER) performance at SNR 1 dB. We calculate compute time as: $T_c = \frac{1}{P_f}\sum_{i=1}^{N_{sub}}(T_a)_i \times (N_a)_i$, where $N_{sub} = 128$ is the number of sub-blocks. $(T_a)_i = 1\mu s$ is the annealing time and $(N_a)_i$ is the number of anneals, of the i^{th} sub-block. BER is computed as in Ref. [9] by considering population samples of $N_a (< 2,000)$ anneals from the entire $2,000$ conducted anneals. The figure shows that higher compute times achieve lower BER. For a target BER of 10^{-4}, compute time required is 250 μs (P_f = 8). With P_f = 100, this time reduces to 20 μs.

BER Performance. Fig. 6 (*Right*) compares HyPD's BER performance head-to-head against SC and SCL decoders. SCL's list size (L_s) is eight, the typical upper limit on list size for 1,024-bit Polar decoder implementations, and HyPD's N_L = 8 (*i.e.,* number of bits in each sub-block §IV-A). The figure shows that HyPD operating at 300 μs compute time outperforms SCL by an half an order of BER magnitude at 1 dB SNR. The performance improvement is a result of QPD decoder exploring all 2^{N_L} decoding paths (*i.e.,* full search) within each sub-block (§IV-A). BER performance can be improved further by sending multiple solutions returned by the QA to classical module, or by increasing the size of partitioned sub-blocks (§IV-A). We note that on current QA hardware, solving sub-blocks with size greater than eight bits leads to poor performance because of accumulated QA ICE errors. Further efforts are needed to enable decoding of longer sub-blocks via QPD, wherein quantum annealing correction may be useful [10]—we leave for future work.

VII. Related Work

Polar codes' fundamental construction and properties are well studied [1], [11] and though proven in theory to be capacity-achieving, their use is limited to short block lengths due to their computationally-complex decoding algorithms. Several studies to this end have proposed efficient decoder architectures based on low-complexity decoding algorithms such as SC [12], SC Stack [13], and belief propagation (BP) [14], whereas HyPD compares favorably in performance against the superior SCL decoding algorithm (§VI). QA machines have been recently used to solve wireless channel decoding problems such as of LDPC codes [9], [15]–[17]. These designs, however, operate on bipartite *Tanner graph* structures, and are not applicable to binary tree structured Polar codes. Quantum Gate-based channel decoding approaches are also being widely investigated, wherein quantum approximate optimization algorithm, quantum search methods, and syndrome-based decoding schemes have been studied [18]–[21]. While we study HyPD from the QA perspective, we note that implementation of same ideas via optimization approaches on Quantum Gate-based devices is also a promising future work direction.

VIII. Conclusion

HyPD is a novel QA-based hybrid classical–quantum Polar decoder design that achieves new levels of performance beyond state-of-the-art SCL decoders at low SNRs of practical interest. While today's quantum technology offers limited number of qubits, restricting the size of problems one can run on such devices, our hybrid decoder design shows how classical computation can be leveraged alongside quantum computation to solve long Polar codes. Our extensive evaluation of HyPD on leading-edge QA hardware provides insights into today's QA hardware performance and benefits with increased qubit numbers. While we acknowledge the practical feasibility of QA processors to be at least tens of years away [5], the ideas we propose here may in the more distant future enable applicability of long Polar codes in NextG cellular wireless traffic channels.

Acknowledgements

This research is supported by National Science Foundation (NSF) Award CNS-1824357. We thank the Princeton Advanced Wireless Systems (PAWS) Group for useful discussions. Support from InterDigital Corporation allowed machine time on a D-Wave Quantum Annealer.

References

[1] E. Arikan, "Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels," *IEEE Transactions on Information Theory*, vol. 55, pp. 3051–3073, 2009.

[2] I. Tal and A. Vardy, "List decoding of polar codes," *IEEE Transactions on Information Theory*, vol. 61, no. 5, pp. 2213–2226, 2015.

[3] 3GPP, "Multiplexing and channel coding," *38.212*, vol. V.15.3.0, 2018.

[4] J. D. L. Ducoing and K. Nikitopoulos, "Quantum annealing for next-generation MU-MIMO detection: Evaluation and challenges," in *IEEE International Conference on Communications*, pp. 637–642, IEEE, 2022.

[5] S. Kasi, P. Warburton, J. Kaewell, and K. Jamieson, "A cost and power feasibility analysis of quantum annealing for NextG cellular wireless networks," *arXiv preprint arXiv:2109.01465*, 2021.

[6] D-Wave, "Technical description of the D-Wave quantum processing unit," *D-Wave Systems User Manual*, pp. 09–1109A–O, 2019.

[7] 3GPP TSG-RAN WG1 Meeting #88 R1-1703106. Nokia, Alcatel-Lucent Shanghai Bell, "Polar design for control channels," 2017.

[8] D-Wave, "D-Wave *minorminer* embedding tool." Github, 2018.

[9] S. Kasi and K. Jamieson, "Towards quantum belief propagation for LDPC decoding in wireless networks," in *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*, pp. 1–14, 2020.

[10] W. Vinci, T. Albash, G. Paz-Silva, I. Hen, and D. A. Lidar, "Quantum annealing correction with minor embedding," *Physical Review A*, vol. 92, no. 4, p. 042310, 2015.

[11] E. Arikan and E. Telatar, "On the rate of channel polarization," in *2009 IEEE International Symposium on Information Theory*, pp. 1493–1495, IEEE, 2009.

[12] K. Chen, K. Niu, and J. Lin, "Improved successive cancellation decoding of polar codes," *IEEE Transactions on Communications*, vol. 61, no. 8, pp. 3100–3107, 2013.

[13] W. Song, H. Zhou, K. Niu, Z. Zhang, L. Li, X. You, and C. Zhang, "Efficient successive cancellation stack decoder for polar codes," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 11, pp. 2608–2619, 2019.

[14] S. Cammerer, M. Ebada, A. Elkelesh, and S. ten Brink, "Sparse graphs for belief propagation decoding of polar codes," in *2018 IEEE International Symposium on Information Theory (ISIT)*, pp. 1465–1469, IEEE, 2018.

[15] N. Ide, T. Asayama, H. Ueno, and M. Ohzeki, "Maximum likelihood channel decoding with quantum annealing machine," in *2020 International Symposium on Information Theory and Its Applications (ISITA)*, pp. 91–95, 2020.

[16] A. D. Sarma, U. Majumder, V. Vaidya, M. G. Chandra, A. A. Kumar, and S. Pramanik, "On quantum-assisted LDPC decoding augmented with classical post-processing," *arXiv preprint arXiv:2204.09940*, 2022.

[17] Z. Bian, F. Chudak, R. Israel, B. Lackey, W. G. Macready, and A. Roy, "Discrete optimization using quantum annealing on sparse Ising models," *Frontiers in Physics*, vol. 2, p. 56, 2014.

[18] T. Matsumine, T. Koike-Akino, and Y. Wang, "Channel decoding with quantum approximate optimization algorithm," in *2019 IEEE International Symposium on Information Theory (ISIT)*, pp. 2574–2578, 2019.

[19] Z. Babar, Z. B. Kaykac Egilmez, L. Xiang, D. Chandra, R. G. Maunder, S. X. Ng, and L. Hanzo, "Polar codes and their quantum-domain counterparts," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 1, pp. 123–155, 2020.

[20] C.-Y. Lai, K.-Y. Kuo, and B.-J. Liao, "Syndrome decoding by quantum approximate optimization," *arXiv preprint arXiv:2207.05942*, 2022.

[21] Z. Babar, P. Botsinis, D. Alanis, S. X. Ng, and L. Hanzo, "Fifteen years of quantum LDPC coding and improved decoding strategies," *IEEE Access*, vol. 3, pp. 2492–2519, 2015.